Social Structures in Software Projects

Soziale Strukture in neuen Softwareprojekten

Dr. Bernhard Scheffold,
(bernhard.scheffold@adinfinitum.de)

Walter Kriha,

(walter@kriha.de)

1

- Wieso beschäftigen sich Software-Entwickler mit sozialen Strukturen?
- Das Leiden am Alten und der Weg zum Neuen
- Beobachtungen: Verhalten, Kategoriensysteme, Architektur, Typen
- Erklärungsversuche
- Verwobenheit sozialer und technischer Strukturen

2

Fragen an gescheiterte Projekte

- Was hat die “Neuen” von den “Alten” unterschieden?
- Welche Modellbildungen waren unannehmbar?
- War der neue Arbeitsstil so anders (basierend auf Kommunikation und gemeinsames Verständnis)?
- Wieso fällt es so schwer, von alten Konzepten Abschied zu nehmen?
- Wie ist der Zusammenhang zwischen technischen Strukturen und Konzepten und sozialen Strukturen?
- Welche Rolle spielt die Arbeitsteilung in der Software-Entwicklung?
- Sind Grundprobleme der Software-Entwicklung wie Zuverlässigkeit, Erweiterbarkeit, Wartbarkeit und Qualität letztlich soziale Phänomene?
- Wieso scheinen sich bestimmte Probleme in Softwareprojekten immer zu wiederholen?

3

Anerkennung sozialer Faktoren wird “legal”

- Design-Pattern-Bewegung
- Soziale Tauglichkeit von Programmiersprachen
- Firmenorganisation als Framework
- Weitere Sichtweise von Software-Architektur: Neue Rollen, Interaktionsmuster, Denkmuster und Kultur der Entwicklung von Software

Formen der Kritik am Alten

Business	Technik	Sozial	Denkmuster
Offen	Offen	Implizit oder gar nicht	kaum
Im Detail	Im Detail, aber oft nicht über den Lebens-zyklus	Kaum Ursachen-forschung	Kein Wissen, Leiden

Explizite, offene Kritik am Alten

- Kosten
- Mangelnde Flexibilität
- Fehler, Qualität
- Schwierige Handhabung
- Aufwendige Installation und Wartung
- Nicht mehr zeitgemäss
- Mangelnde Kapselung macht Änderungen schwierig und gefährlich
- Keine Reuse der Software

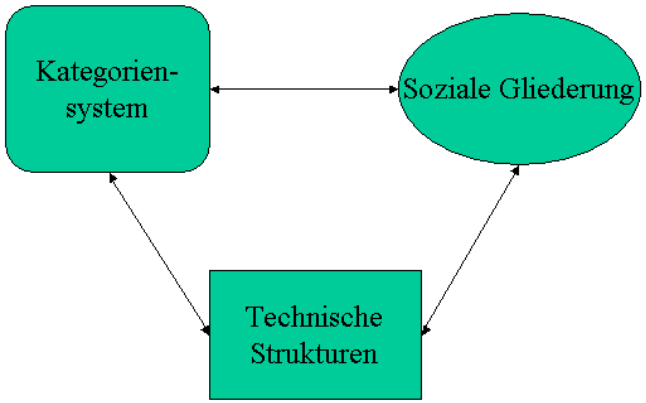
Implizite, versteckte Kritik am Alten

- Verwalter des alten Systems sind arrogant
- Sie wollen keine Veränderungen
- Keine benutzerfreundliche Organisation
- Gängelung statt Unterstützung
- Undurchschaubare, zirkuläre Argumente
- Technik dominiert Business

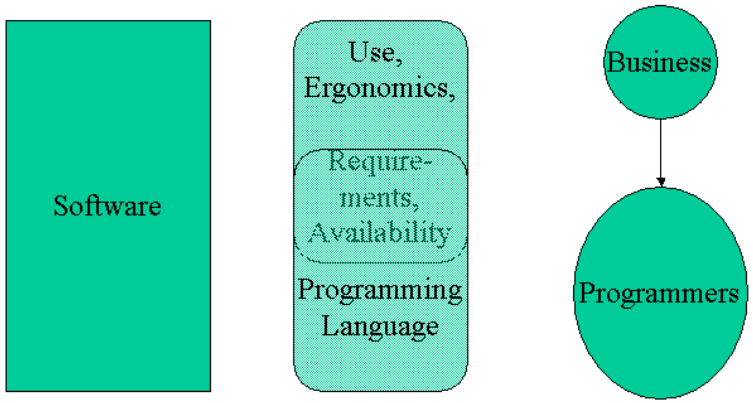
Kritik alter Denkmuster
- Hilflosigkeit -

- Die Auseinandersetzung wird gescheut
- Wenn sie stattfindet, ist sie persönlich schmerzhaft und aufreibend
- Erfolg (sprich: Aufweichen der Denkmuster) ist minimal
- Rückfälle in alte Denkmuster
- Einzelne Personen schaffen den Sprung in die neuen Denkbilder
- Prinzip Hoffnung und neue Leute.

Grundbausteine der Software-Entwicklung als sozialer Prozess

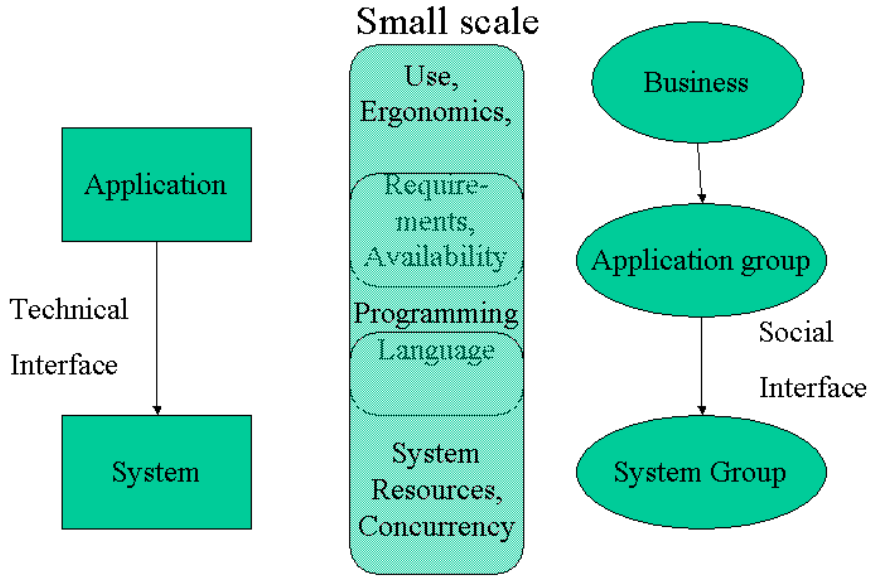


First SW-Architecture Model



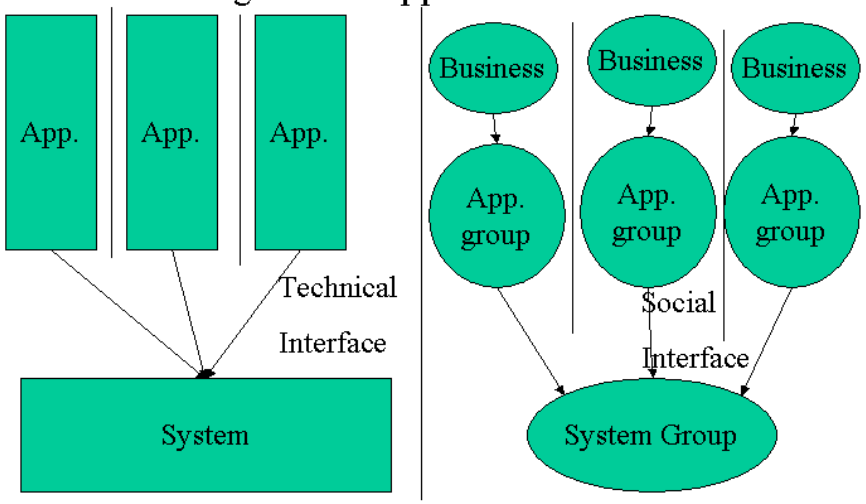
A piece of software gets downloaded to special hardware. It contains system and application. No decomposition of software or programming

Base-Model SW-Architecture



Base-Model SW-Architecture

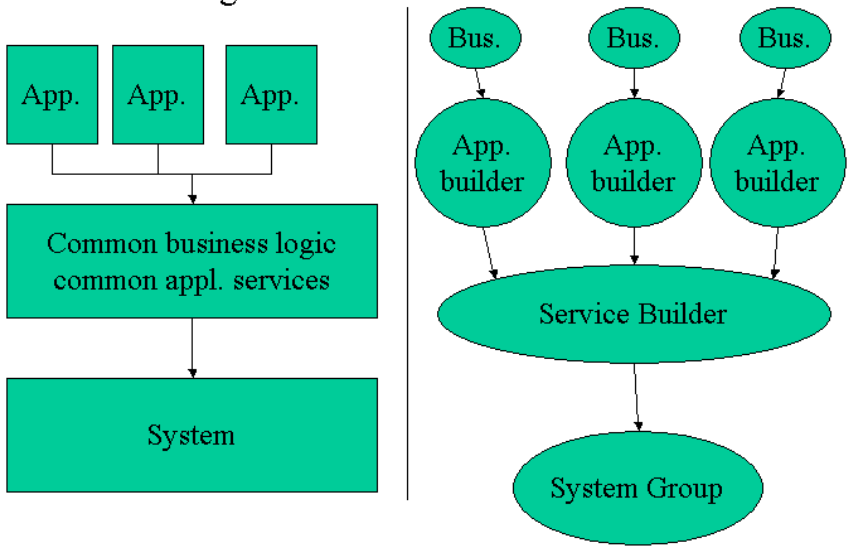
Large scale: application tower



12

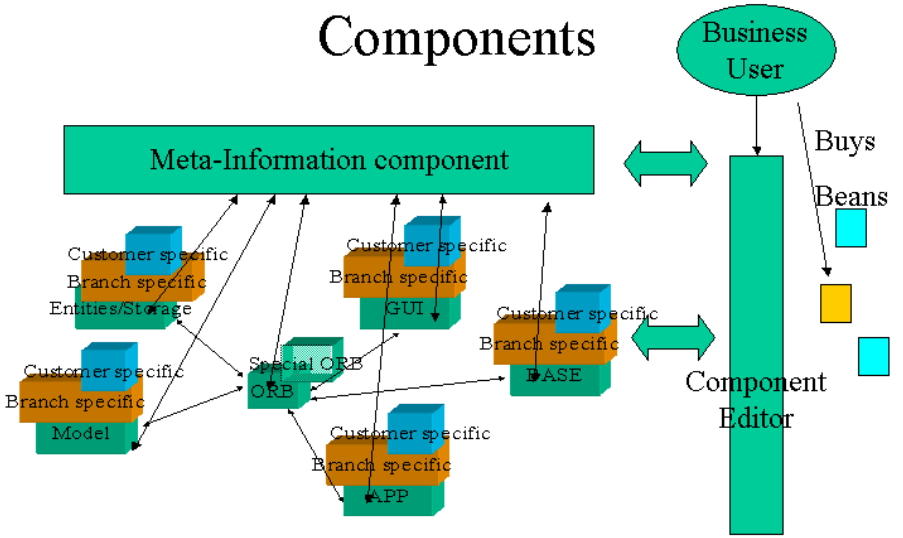
3-Tier-Model SW-Architecture

Large scale: common services



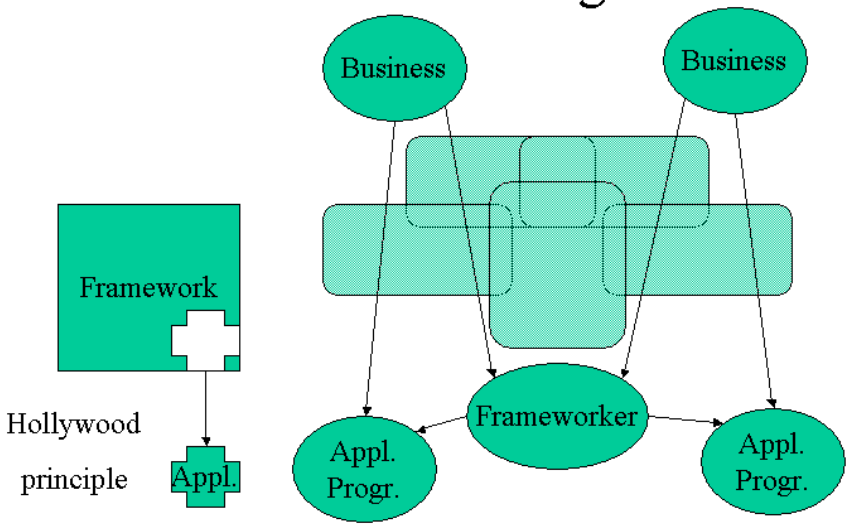
13

Framework Architecture with Components



14

Frameworking



15

New roles for component models

Technical roles

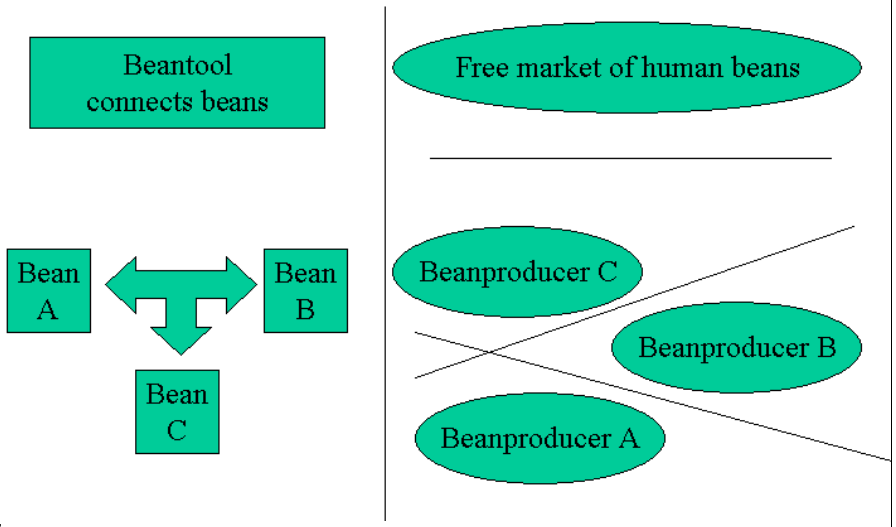
- Business/App. Architect
- Business Object Architect
- Component Developer
- System Object Architect
- System Architect

Business roles

- Business Project Manager
- IT Project Manager
- Component Owner

Simple Component Architecture

Minimizes social interfaces



Social reasons why new architectures fail

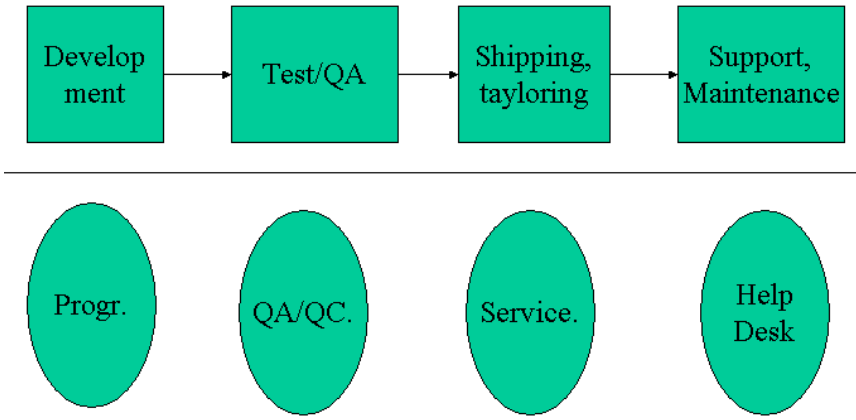
Old app. Roles/kategories

- Oriented towards one specific business case, must be done quickly
- expectation of fixed API to system
- technical competence for applications is within the app. Programming groups

New app. Roles/kategories

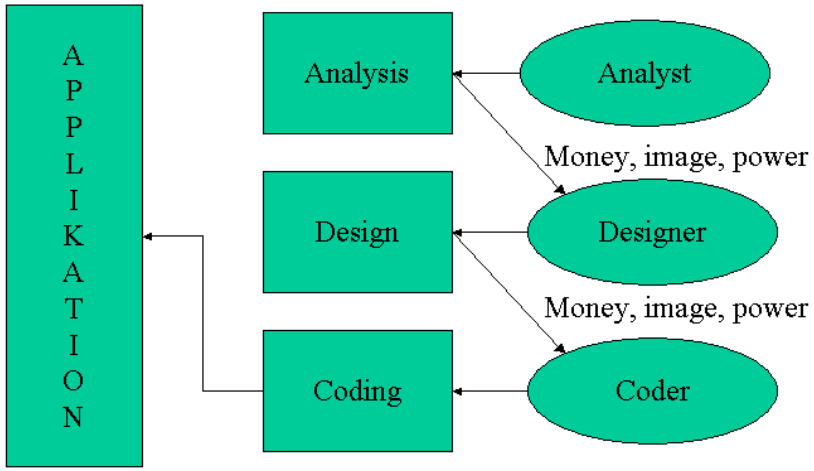
- building generic parts
- reuse over speed
- system is changeable, needs arch. Knowledge
- Role threatened by enabling technology: business users can build the final applications

Lifecycle view



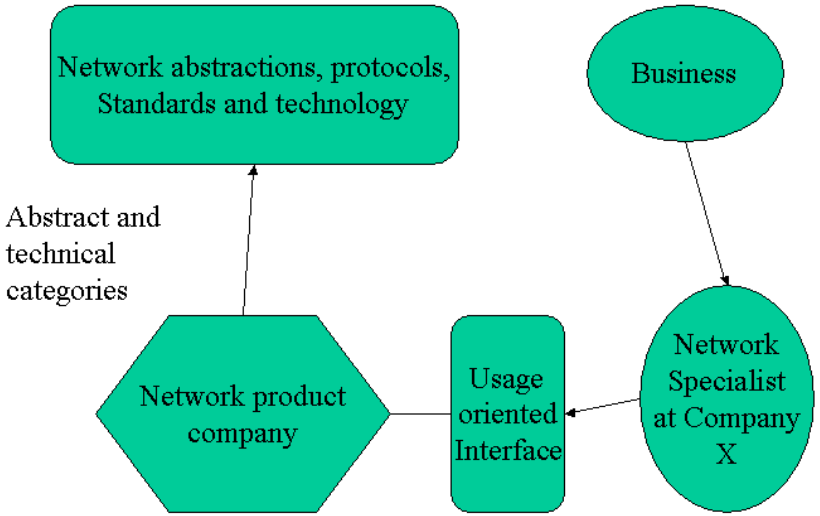
Reliability, Quality and Extendability show up as problems not in development but in other groups.

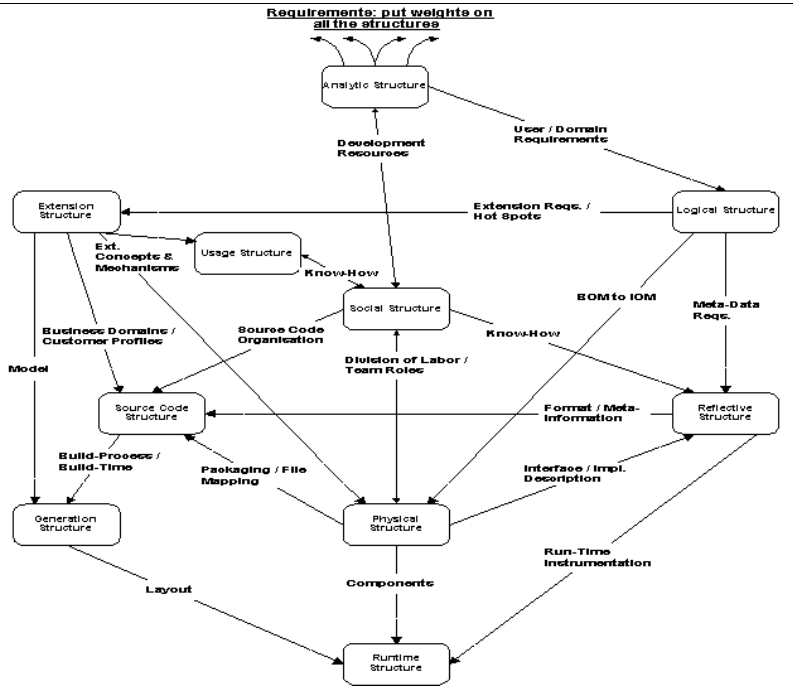
Waterfall Development view



Who takes care of performance, reuse etc.? Who sees the whole?

How Networking becomes "Novell"





Some requirements of successful projects

- Multi-dimensional decomposition of architecture
- Projection of technical and social architecture over time
- Make category systems explicit: no single "right" view
- Expect multiple and changing category systems. Architecture must support those
- Beware of "mapping" approaches. They try to reduce complexity to just one category system and fail in reality
- Minimize social interaction using framework technology
- Maximize social interaction by separating social interfaces from technical interfaces
- Micro level of coding: Make the connection between complexity and abstraction visible and socially understandable.